EBM + Normalized Langevin + LP Worker

Improvement Roadmap (detailed version)

Théotime Coudray

December 24, 2025

Abstract

This document summarizes, in a practically motivated priority order, the most promising improvements for a surrogate MILP UC/flexibility solving pipeline built on an *Energy-Based Model* (EBM), a *Normalized Langevin* sampler, and an *LP worker*. The goal is to reduce the *tail* of pathological scenarios (extreme gaps) by improving: (i) the temporal expressivity of the energy function, (ii) alignment with the true LP cost, and (iii) robustness and operational reliability (diagnostics, guardrails).

Overview: recommended order

1. **Zone-level** embeddings (spatially relevant context)
2. Energy **over zone temporal trajectories** (intertemporal dynamics)
3. **Margin-based** loss (contrastive stabilization / outlier reduction)
4. **Cost-aware** training (alignment with real economic impact)
5. Explicit **soft constraints** as energy terms (physics priors)
6. Penalizing **fragility** (flat minima / stable solutions)
7. Energy **self-diagnostics** (confidence score, fallback)
8. **Causal regularization** (generalization / directional coherence)
9. **Hard negatives** via Normalized Langevin during training (phase 2)
10. **Sampler tuning** (noise, steps, restarts) (practical optimization)

1 Preliminaries: what we are trying to fix

A standard EBM learns an energy $E_\theta(u \mid h)$ that should be *low* on observed binary configurations u (commitments/modes) coming from the MILP oracle, and *high* on negative configurations. The main risk in UC/flexibility problems is not the mean gap, but the *tail*: a handful of scenarios where a small number of badly placed bits in time triggers huge costs at the LP worker (VOLL, forced imports, ramp/storage inconsistencies). The improvements below primarily aim to:

- capture intertemporal dependencies (storage, ramping, startups),
- align learning with the true economic impact of decisions,
- make the pipeline robust (uncertainty detection, stable solutions).

2 1) Move to zone-level embeddings

1. Zone-level embeddings

Goal. Provide the EBM with a context h that preserves local tensions (congestion, VRE, demand, local flexibility).
Why. A "averaged" national embedding collapses structure: two scenarios with the same national aggregates can be radically different at the zone level (local deficits, limited imports, saturated local storage). An EBM conditioned on an overly global context tends to generate "average" decisions and misses the corners of the feasible polytope.
Implementation.

- Extract $h_{z,t} \in \mathbb{R}^D$ for each zone z and time step t (e.g., via your multiscale encoder).
- In the `DataLoader`, attach each binary variable $u_{z,t,\cdot}$ to *its* zone embedding: broadcast $h_{z,t}$ only to nodes/variables belonging to that zone.
- If u is "per zone and time", you can represent each pair (z, t) as a decision node.

What to measure.

- Reduction of extreme gaps (90/95/99th percentiles).
- Oracle cost vs LP-worker cost correlation (log-log).
- LP feasibility rate and sensitivity to "OOD" scenarios.

Pitfalls.

- Aligning zone/time identifiers between embeddings and binary dictionaries is a major source of bugs.
- If your EBM remains a "Deep Sets" model without temporal structure, gains exist but quickly plateau.

Expected gain. Better spatial conditioning and fewer locally inconsistent decisions.

3 2) Inject temporal dynamics: energy over zone trajectories

2. Energy over temporal trajectories (zone)

Goal. Make the EBM learn intertemporal strategies (e.g., *discharge now to survive a later peak*).
Why. Pathological scenarios often come from poor *timing*: a correct bit at time t becomes catastrophic at $t+k$ (SOC, ramp limits, DR availability). An "instantaneous" energy summed over independent variables cannot represent these dependencies.
Implementation. (simple → advanced options)

1. **Local window** : define energy on $(t-1, t, t+1)$ per zone: $E = \sum_{z,t} e_\theta(u_{z,t-1:t+1}, h_{z,t-1:t+1})$.
2. **Full trajectory** : $E = \sum_z E_\theta(u_{z,1:T}, h_{z,1:T})$ where E_θ is a small GRU/TCN/light Transformer per zone.
3. **Hierarchy** : add a national term $E^{\text{nat}}(u, h^{\text{nat}})$ for global decisions (imports, balance constraints), while keeping the zone term.

What to measure.

- Fewer ON/OFF toggles and fewer unnecessary startups.
- Battery/pumped cycling distributions vs oracle (simple statistics).
- "Best-of-k" curve: as k increases, you approach oracle without blowing up.

Pitfalls.

- Watch memory if you concatenate zone×time×features naively.
- If you do not control energy scaling, Langevin can diverge (normalization helps).

Expected gain. Captures intertemporal dynamics: major reduction of the outlier tail.

4 3) Margin-based loss (margin ranking) to stabilize training

3. Margin-based loss instead of pure gap

Goal. Prevent training from being satisfied with a small mean separation while keeping outliers.
Why. The mean gap $E_{pos} - E_{neg}$ can be small without guaranteeing robust separation. A margin loss enforces a minimum separation m: you do not "relax" until $E_{neg} \geq E_{pos} + m$.
Implementation.

$$\mathcal{L} = \max(0,\, m + \mathbb{E}[E_{pos}] - \mathbb{E}[E_{neg}]) + \lambda\,\text{Reg}$$

- Choose m (e.g., 1.0, 2.0) and possibly a schedule (it can increase with epochs).
- Keep a regularizer on the energy scale (L2 on E or on weights).

What to measure.

- Energy stability (no explosion) and improved separation on validation.
- Reduction of LP-cost outliers.

Pitfalls. Too large m can slow learning; too small reduces to pure-gap training. **Expected gain.** Stabilization and fast reduction of catastrophic cases.

5 4) Cost-aware training (align energy with the economic objective)

4. Cost-aware training

Goal. Ensure costly mistakes are penalized far more than benign mistakes.
Why. In UC/flexibility, a bit on thermal commitment at the right time can cost $\times 10^2$–$\times 10^4$ more than a DR bit. Without weighting, the EBM can reduce loss by fixing easy bits and leave rare but ruinous mistakes.
Implementation. (pragmatic)

- Define weights w_i by variable type and timestep.
- Example: w proportional to `startup_cost`, `VOLL`, or a proxy (shadow price / dual).
- Inject w into the loss: $\sum_i w_i \, e_\theta(u_i, h)$ or weight samples/batches.

What to measure.

- Lower 99th percentile of gaps.
- Better monotonic relationship between EBM energy and LP cost.

Pitfalls. Over-aggressive weighting may hurt the mean; the target is the tail. **Expected gain.** Reduced "explosive" scenarios with very high costs.

6 5) Explicit soft constraints as energy terms

5. Add dedicated energy terms (physics priors)

Goal. Avoid forcing the EBM to relearn trivial rules; steer the sampler away from toxic regions.
Why. Some inconsistencies are well known: rapid toggles, simultaneous charge/discharge, DR activated outside windows, etc. Penalizing them directly stabilizes sampling and reduces LP failures.
Implementation.

- Add $E(u) = E_{\text{data}}(u) + \lambda_1 V_{\text{toggle}}(u) + \lambda_2 V_{\text{stor}}(u) + \cdots$
- V_{toggle}: penalty on $|u_t - u_{t-1}|$ for commitments (or on the number of switches).
- V_{stor}: penalize simultaneous charge and discharge, or overly fast cycles.

What to measure.

- Lower LP infeasibility rate.
- Fewer toggles and fewer "non-physical" behaviors.

Pitfalls. Too many soft constraints $\Rightarrow$ overly rigid model; tune λ's. **Expected gain.** Very strong ROI against pathological scenarios.

7 6) Penalize fragility: prefer "flat" minima

6. Penalize fragility (robustness)

Goal. Favor stable solutions: small perturbations should not make the LP catastrophic.
Why. Narrow minima (energy highly sensitive to one flip) are dangerous: they can look good in energy yet be poor in true economic cost.
Implementation.

- Penalize gradient norm: $\lambda \|\nabla_u E\|^2$ (on a continuous relaxation).
- Or penalize variance: evaluate $E(u + \epsilon)$ for a few perturbations and penalize the variance.

What to measure. Tail of gaps, stability of best-of-k candidates, sensitivity to random flips. **Pitfalls.** Computational overhead if too many perturbations; keep 2–4 per batch. **Expected gain.** Strong reduction of extreme failures.

8 7) Energy self-diagnostics (confidence score)

7. Self-diagnostics: estimate confidence

Goal. Know when the pipeline should fallback (multi-restart, partial MILP, etc.).
Why. Even a strong EBM will face OOD cases. A confidence score prevents catastrophic decisions in production.
Implementation.

- Add an auxiliary head $c_\phi(h, u)$ predicting local energy variance under perturbations, or a "margin" score $(E_2 - E_1)$.
- Define a rule: low confidence $\Rightarrow$ more restarts / partial exact solve.

What to measure. Calibration: risk-coverage curve (accept fewer cases, but more reliable ones). **Pitfalls.** The head must not learn trivial proxies (e.g., graph size); regularize accordingly. **Expected gain.** Industrial robustness and controllability.

9 8) Causal regularization

8. Causal regularization

Goal. Improve generalization by enforcing directional dependencies (t $\rightarrow$ t+1).
Why. Some relationships must be directional (SOC, ramps, commitment). Without an inductive bias, the EBM may capture brittle correlations.
Implementation.

- Penalize temporal order violations (e.g., decisions depending on future information).
- Add constraints/regularizers on causal features (state $\rightarrow$ decision).

What to measure. OOD robustness (extreme climate stress, atypical VRE profiles). **Pitfalls.** Can be hard to define cleanly; start from obvious physical rules. **Expected gain.** Better generalization (especially out of distribution).

10 9) Hard negatives via Normalized Langevin during training (phase 2)

9. Hard negatives: Langevin in the training loop

Goal. Make the EBM discriminative against realistic, challenging negatives.
Why. Bernoulli negatives are too far; the EBM learns to separate easy noise. Langevin provides candidates close to the modes.
Implementation. (careful)

- Warm-up with simple negatives (epochs 1–K).
- Then, 1 batch out of k: generate u_{neg} with a *short* Normalized Langevin run (few steps), and use it as negatives.

What to measure. Energy stability and better performance on outliers. **Pitfalls.** If the sampler is too strong too early: collapse / cheating. **Expected gain.** Higher precision on difficult cases.

11 10) Sampler tuning (noise, steps, restarts)

10. Sampler tuning

Goal. Improve the speed/quality trade-off and reduce narrow minima.
Why. More iterations is not always better; too much precision can trap you in bad minima.
Implementation.

- Multi-restart (best-of-k) with few steps rather than one long run.
- Stronger initial noise, smoother annealing.
- "Block noise": flips by temporal/zone blocks.

What to measure. Quality-vs-time curve (latency) and tail of gaps. **Pitfalls.** Too much noise $\Rightarrow$ exploration without convergence; calibrate. **Expected gain.** Practical improvements without changing the model.

12 Conclusion

The priority is to make the energy function *dynamic* and *cost-aligned*: (i) zone embeddings, (ii) energy over zone temporal trajectories, (iii) margin loss, (iv) cost-aware training, then robustness and industrialization. The objective is not only to improve the median gap, but to cut the tail of catastrophic scenarios (e.g., cases similar to scenario 00643).